ワンチップ・マイコンを利用した回路設計基礎技術の修得（Ⅱ）

専門研修受講者　　岡井 善四郎，酒井 孝則，本堂 義記（第三技術室），
田畑 功（第二技術室），小川 勇治（第一技術室）

1．はじめに

昨年の研修では、PICを用いた簡単なデジタル温度計・電圧測定器の設計・製作を通して、PICの基礎を学習した。今年度は、はじめにPICによるLCD制御を重点にC言語によるプログラミング技法について復習を行った。次に「PIC応用ガイドブック」の輪読を行い、プログラミングの基礎となるVHDL（Very high speed integrated circuit　Hardware Description Language）文法を中心に、パソコンを用いてのプログラム開発方法を修得した。実習では、開発に必要なダウンロードケーブル（HUMANDATAのキット）、汎用CPLDテストユニットの製作を行い、簡単なデモプログラムを作成し、その動作を確認した。

2．ＣＰＬＤについて

PICには入力したデジタル信号をソフト的に論理処理して出力する機能があるが、PICのみでは処理速度や機能が不足する場合が多い。このような場合、PLD(Programmable Logic Device)と呼ばれる素子をPICに接続することで、より高速動作、高機能な回路設計・製作が可能となる。このIC素子は、VHDLという標準開発言語でデジタル回路を自由に構成できるもので、大きく分けて、FPGA（Field Programmable Gate Array）とCPLD（Complex Programmable Logic Device）がある。ここでは扱いやすく入手が容易なXilinx社のXC9500シリーズCPLDを使用した。

CPLDは、デジタル回路の内容をフラッシュメモリーに記憶する方式のため、回路設計通りの動作をするまでプログラムを何回でも書き換え可能である（10,000回程度）。また、ISP（In-System Programming）と称する書き換え用のピンが用意されているため、オンボードでの書き込みが可能である。更に、PLCC（Plastic Leaded Chip Carrier）ソケット（ピン間隔が0.1インチ）を使用しているため、ユニバーサルプリント基板を使った手軽な電子回路の製作に向いている。

2．1　素子の構造と動作

内部は図1のように、①入出力ブロック部（外部ピンとの入出力バッファでプログラムによりいずれかの機能ブロックと接続されて動作する）、②高速接続スイッチマトリクス部（プログラムの内容に従い機能ブロックと外部ピンとの接続やブロック同士を接続する）、③機能ブロック部（18個のマクロセルが内蔵されており、プログラムによりいろいろなロジック回路を構成する）を中心に、その他JTAG制御部、インーシステムプログラム制御部から構成されている。

高速接続マトリクス部には全てのI／Oポートからの入力信号と機能ブロックの出力信号が接続されており、プログラムで指定された信号が機能ブロックへ伝えられる。機能ブロックは図2のような構成で、ここに入力される36本の入力はプログラムANDアレイにより、真／偽の信号に分けられ72種類の信号になる。機能割付部では、プログラムに従いこれらの信号を組み合わせ必要な論理信号として、図3のマクロセルに供給する。

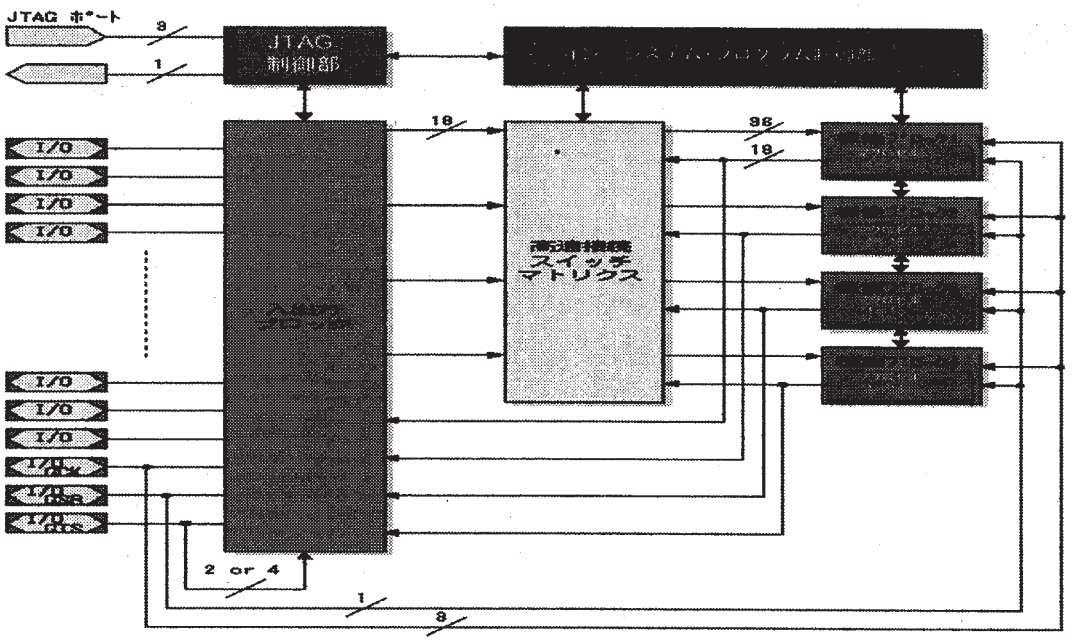

図1　XC9500 のブロック図[1]

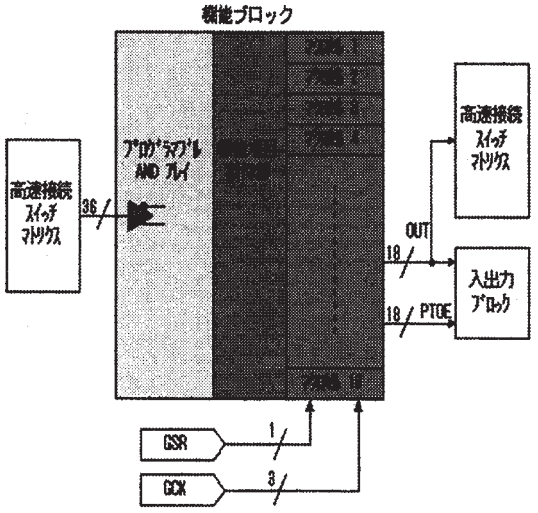

図2　機能ブロック部[1]

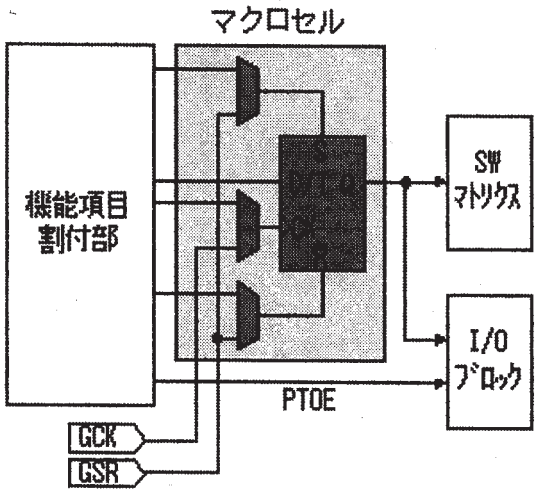

図3　マクロセル[1]

さらにマクロセルの入出力を、高速スイッチマトリクスで他のマクロセルに接続したり他のブロックに接続したりして組み合わせることで、目的の機能が得られる。

2．2 VHDL の文法と概要

CPLD へ書き込むデジタル回路データの作成に VHDL 言語が使われている。この研修では VHDL 言語を使用し、直接論理合成が可能なレベルの RTL（Register Transfer Level）記述を使用する。VHDL の書式構成は、図 7 のように、「entity」，「architecture」，並びに各種の宣言部から成り立っている。各部の役割を見てみるとパッケージの指定：ライブラリパッケージの指定，各種の演算子や標準関数などの定義がなされる。標準ライブラリパッケージ（library IEEE）の指定は必須である。演算子を使うときには更に別のパッケージが必要である。

① entity：外部との入出力インターフェースを定義する部分で、外部との入出力ピンを port として定義する。その他、属性宣言、ジェネリック宣言などを行う。

② architecture：実際の内部の動作を記述する部分で順次処理文、同時処理文などを C 言語に似た書式で記述する。

3．製作

CPLD に PC ポートを通して書き込みするためのダウンロードケーブル（XCKIT, HUMANDATA 社）を製作した。次に、輪読した「PIC 応用ガイドブック」に掲載されている（汎用テストユニット）を、付録の回路図、パターン図を利用して製作した。製作に当たりユニット基板上に 10P の JTAG ソケットを用意した。図 4 に製作した汎用テストユニットの回路図を、図 5 に製作したダウンロードケーブルと汎用テストユニットを示す。

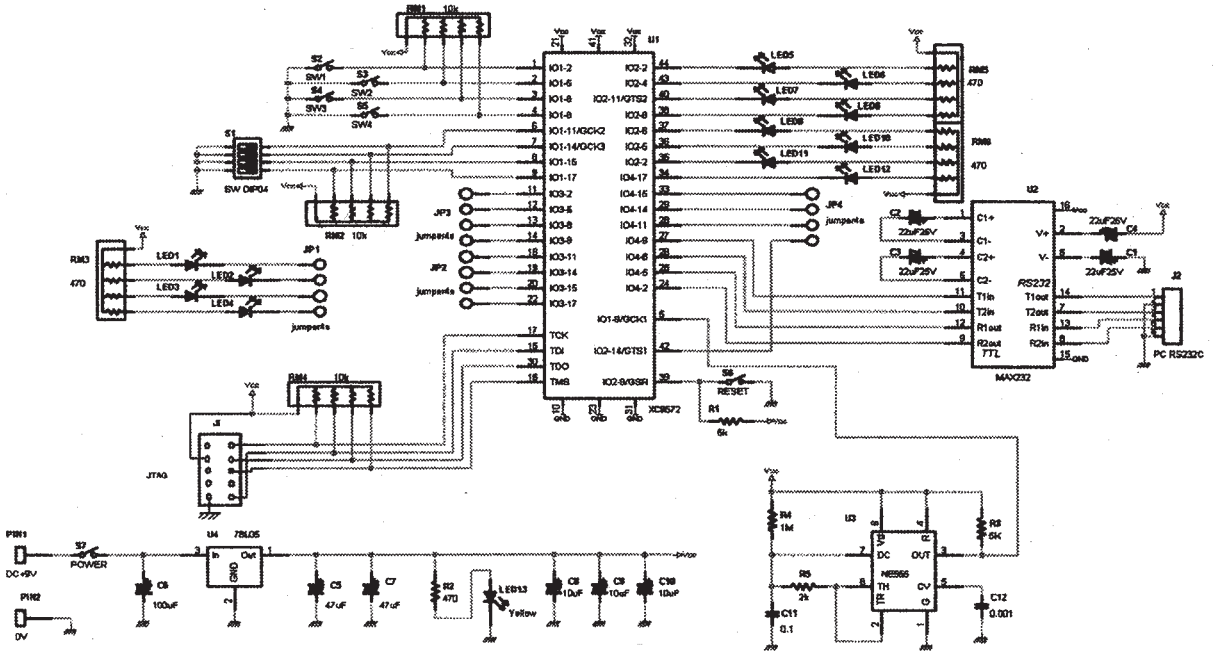

図 4　汎用テストユニット回路

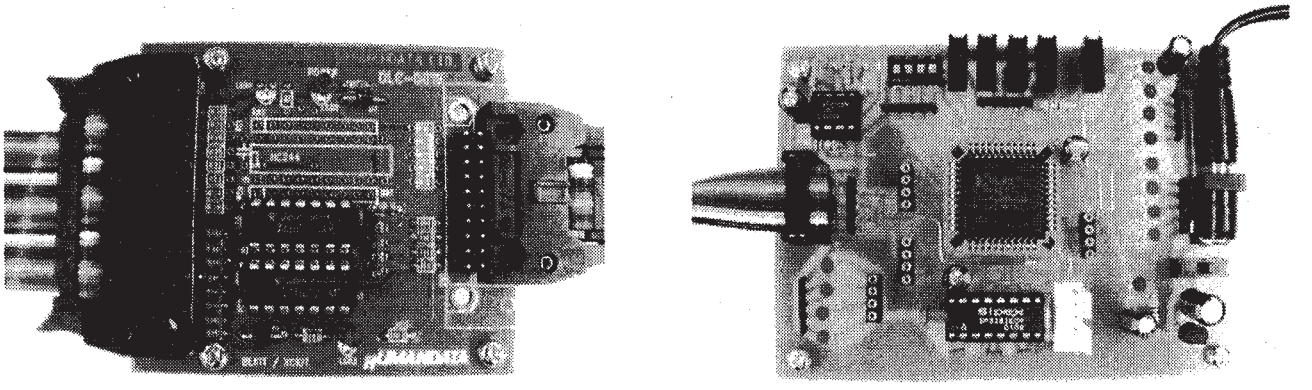

図 5　製作したダウンロードケーブル・汎用テストユニット

4．ＣＰＬＤテストユニットを用いた実習

4．1 LED 点滅プログラムの作成

製作したダウンロードケーブル、テストユニットを用いた LED 点滅デモプログラムを作成した（図 6）。プログラムの開発には、Xilinx 社が無料で提供している「WebPACK ISE」Ver.5.1.03i を用いた。このプログラムは、ホームページ[5]上で公開されている「20 進アップダウンカウンタ」プログラムをベースに作成した。

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--  Uncomment the following lines to use the declarations that are
--  provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;


entity led_demo is
  port ( SW_1,SW_2,CLK : in std_logic;
              LED : out std_logic_vector(7 downto 0)
       );

attribute pin_assign : string;
attribute pin_assign of LED    : signal is   "34,35,36,37,38,40,43,44";
attribute pin_assign of CLK    : signal is   "5";
attribute pin_assign of SW_1   : signal is   "1";
attribute pin_assign of SW_2   : signal is   "2";

end led_demo;

architecture RTL of led_demo is

signal CLK_2      : std_logic_vector(2 downto 0);  -- クロック 4 分周
signal DCLK       : std_logic;                     -- 遅延クロック
signal CNT        : std_logic_vector(3 downto 0);  -- カウント数
signal SW_STATE : std_logic_vector(1 downto 0);    -- スイッチの状
態

begin

-- クロック分周
  process begin
    wait until CLK'event and CLK = '0';    -- クロックの立下がり
    CLK_2 <= CLK_2 + '1';
  end process;

  DCLK <= CLK_2(2);          -- 遅延クロック生成
  SW_STATE <= SW_1 & SW_2; -- スイッチの状態

-- 遅延クロック同期カウンタの動作
  process begin
    wait until DCLK'event and DCLK = '1'; -- 遅延クロックの立上が
り
    case SW_STATE is
      when "00" =>                    -- SW_1,SW_2 ON
        CNT <= "0000";                -- カウント初期化

      when "01" =>                    -- SW_1 ON ,SW_2 OFF
            if ( CNT = "1111" ) then    -- 最大数のとき
        CNT <= "0000";                -- カウント初期化
            else
        CNT <= CNT + "0001";          -- カウントアップ
      end if;

      when "10" =>                    -- SW_1 OFF, SW_2 ON
            if ( CNT = "0000") then     -- 最小数のとき
        CNT <= "1111";                      -- 最大数にセット
      else
        CNT <= CNT - "0001";          -- カウントダウン
      end if;

      when others => null;            -- 何もしない
    end case;
```

図 6　LED 点滅プログラム

```
end process;

  -- カウンタの値を LED に表示
  process ( CNT ) begin
    case CNT is
      when "0000" => LED <= "11111110";   -- LED5 を点灯
            when "0001" => LED <= "11111101";   -- LED6 を点灯
            when "0010" => LED <= "11111011";   -- LED7 を点灯
            when "0011" => LED <= "11110111";   -- LED8 を点灯
            when "0100" => LED <= "11101111";   -- LED9 を点灯
            when "0101" => LED <= "11011111";   -- LED10 を点灯
            when "0110" => LED <= "10111111";   -- LED11 を点灯
            when "0111" => LED <= "01111111";   -- LED12 を点灯
            when "1000" => LED <= "00001111";   -- 赤 4 つを点灯
            when "1001" => LED <= "11110000";   -- 緑 4 つを点灯
            when "1010" => LED <= "00001111";   -- 赤 4 つを点灯
            when "1011" => LED <= "11110000";   -- 緑 4 つを点灯
            when "1100" => LED <= "00000000";   -- すべて点灯
            when "1101" => LED <= "11111111";   -- すべて消灯
            when "1110" => LED <= "00000000";   -- すべて点灯
            when "1111" => LED <= "11111111";   -- すべて消灯
            when others => LED <= "XXXXXXXX";   -- 不定
    end case;
  end process;
end RTL;
```

【全体構造図】

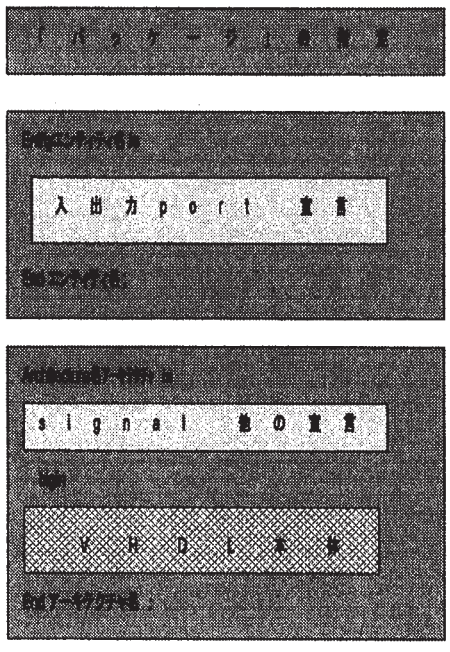

図 7　VHDL 全体構造図

プログラムは、図7のVHDL全体構造図に示すようにライブラリパッケージの指定から始まる。ここでは基本関数ライブラリのほか、符号なし演算関数、算術演算関数用ライブラリを指定している。「. all」を付加することで、内部で定義されているすべての関数を使用可能としている。

Entity節では、外部との入出力ピンをportとして定義している。また属性宣言で以下のピンにアサインしている。(LEDの出力を34～38、40、43、44番ピン、CLKを5、SW 1を1、SW 2を2番ピン)。

Architecture節では、最初にsignal（信号宣言）文で、内部で使用する信号の定義をしている。CPLDで実際に実装する回路は同期回路とする。これはクロック信号に同期して動作するため、ハザードの影響を避けられる。具体的にはprocess文内でクロックの立ち上がりか、立下りで動作するように記述する。クロックのエッジ検出にはwait until文を使っている。次にcase文で多方向分岐を指示し、カウント数に応じて(0～16)LEDの点灯を制御している。

4．2　CPLDへのプログラム書き込み

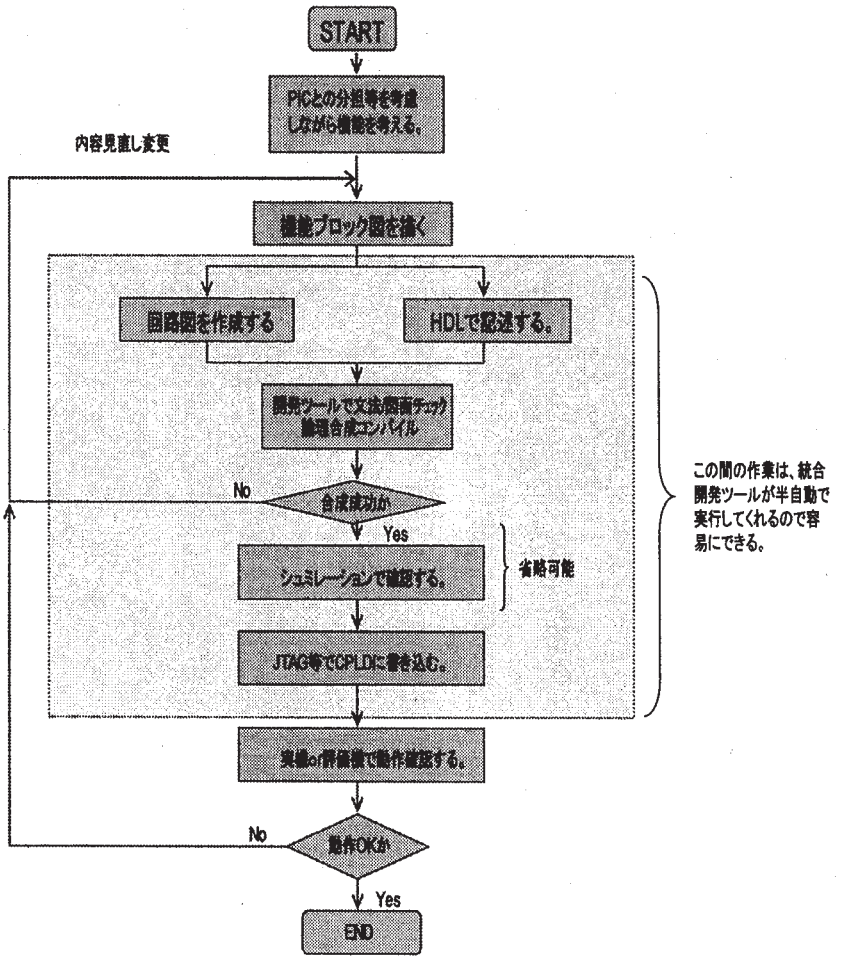

図8　CPLD開発フロー

プログラム作成後、CPLDへの書き込みは、図8に示した流れに従い以下の手順で行う。

① VHDLプログラムの入力後、構文などのチェックを行う。

② 合成。VHDLの構文が正常であればコンパイルして、次の工程で扱いやすいコンフィギュレーションデータ形式に変換する。

③ 配置配線。これはデハイスに書き込むために、どのように内部を接続し、どうマクロセルを使うかを考える工程であり、指定したデバイスに入りきるかどうかのチェックをする。Project Navigatorの Process for Current Source 窓の Implement Design をダブルクリックするだけで自動実行され、成功すれば正常完了の緑のチェックマークが付く。

④ デバイスへ書き込み。生成されたデータを実デバイスに書き込み動作可能とする。

⑤ 動作確認。実際のデバイスで動作させて、プログラム通りの動作が行われるか確認する。

以上の実習を各自が行いLED点滅デモプログラムの動作確認を行った。

5．まとめ

本年度は、PIC プログラミング（C 言語）を復習し、LCD 制御について基本的事項を修得した。次に CPLD について輪読によりその概要を修得した。ダウンロードケーブル、汎用 CPLD テストユニットを製作し、Xilinx 社の「Web Pack ISE」CPLD 統合開発環境を利用して、VHDL 言語でのプグラム作成から、論理合成によりコンフィギュレーションデータに変換し CPLD へ書き込む一連の実習を行い、CPLD を使用した回路設計の基本を修得した。

6．今後の課題

① PIC マイコンと CPLD 素子を利用した高速デジタル信号処理システムを構築する。

② C 言語が苦手な受講者については更なる C プログラミング研修を行う。また、独自の CPLD プログラム開発が行えるよう VHDL 言語を引き続き学習する。

参考文献

1) http://www.hobby-elec.org/cpld2_1.htm

2) 宮崎　仁 ；C 言語とプログラミングの基礎をつかむ，トランジスタ技術、CQ 出版社，Dec (1992)

3) 後閑 哲也；電子工作のための PIC 活用ガイドブック，技術評論社（2000）

4) 後閑 哲也；電子制御のための PIC 応用ガイドブック，技術評論社（2002）

5) http://flex.ee.uec.ac.jp/japanese/riron/vhdl/vhdl

研修日程

専 門 研 修 実 施 日 程

実 施 日	実 施 時 間	主 な 研 修 実 施 内 容
9月　2日（月）	10:15～11:45	本年度専門研修日程および内容等の検討
10月　7日（月）	9:30～11:45	ＬＣＤ駆動用プログラムの作り方（昨年度復習）
10月28日（月）	9:30～11:45	ＬＣＤ駆動用プログラムの検討（昨年度復習）
11月12日（火）	9:30～11:30	ＬＣＤ駆動用デモプログラムの検討（昨年度復習の完了）
11月25日（月）	9:30～11:50	PIC 応用ｶﾞｲﾄﾞﾌﾞｯｸの輪読、実習回路（CPLD 汎用ﾃｽﾄﾕﾆｯﾄ）検討
12月　2日（月）	9:30～11:50	PIC 応用ｶﾞｲﾄﾞﾌﾞｯｸの輪読、実習回路部品の検討
12月　9日（月）	9:30～11:45	PIC 応用ｶﾞｲﾄﾞﾌﾞｯｸの輪読、実習回路部品の決定
12月16日（月）	9:30～11:40	実習回路部品の確認、ﾀﾞｳﾝﾛｰﾄﾞｹｰﾌﾞﾙの製作方法
1月　8日（水）	9:30～11:00	実習回路の検討と各部品の取り扱い方法
1月20日（月）	9:30～11:30	実習回路用基板のｱｰﾄﾜｰｸ技法の検討
2月　3日（月）	9:30～11:30	実習製作回路の進展状況確認
2月17日（月）	9:30～11:15	実習製作回路の動作確認および不具合の検討
3月　3日（月）	9:30～12:00	技術報告集原稿内容の検討と分担
12月17日～翌年2月28日（30時間程度）		ﾀﾞｳﾝﾛｰﾄﾞｹｰﾌﾞﾙ、汎用ﾃｽﾄﾕﾆｯﾄ回路製作およびﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ製作の自主研修